# Software work in mathematical biology

Maxie Dion Schmidt

maxieds@gmail.com
http://people.math.gatech.edu/~mschmidt34
https://github.com/maxieds

Sandia National Labs
Technical Presentation
Spring 2022

## Applications of RNA research

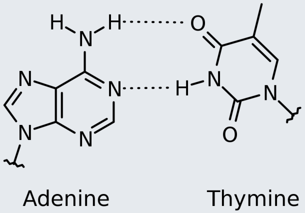
- Ribonucleic acid (RNA) sequencing has been utilized in many aspects of cancer research and therapy
- mRNA vaccines have elicited potent immunity against infectious disease targets in animal models of several key viruses in the past few years
- Research on RNA may eventually play a central role in diagnostics, therapeutics and research in medical applications
- As the cellular roles for RNA molecules continue to grow, so does the importance of gaining functional insight from structural analyses

## RNA sequence basics

- RNA is a single-stranded molecule similar to DNA; it carries messenger instructions to the double-stranded DNA that encodes genetic instructions required for life processes
- A strand of RNA has a backbone comprised of alternating sugar (ribose) and phosphate groups
- Each sugar has one of four base types attached to it: Adenine – **A**, uracil – **U**, cytosine – **C**, or guanine – **G**
- Each of the **A–U–C–G** bases can fold to form bonds with one another

---

**Example:** *The adenine-thymine (AT) Watson-Crick base pair (*`WP/Base_pair`*):*



## RNA secondary structures

- A RNA for an organism is defined by the 1D base sequence and can have more than one secondary and tertiary structure by folding
- While obtaining the 1D sequence information is now relatively easy via sequencing, characterizing 3D molecular conformations is still comparatively hard
- Hence, understanding the 2D secondary structures, i.e. the intra-sequence base pairing, remain a crucial component of ribonomics research
- RNA folding prediction programs (e.g., GTFold or RNAFold) can take a base sequence and use probabilistic algorithms to generate the secondary structure
- The secondary structures generated by such computational procedures can be influenced by certain pairing penalties and by environmental constraints like a thermodynamic model specification

## What is mathematical biology?

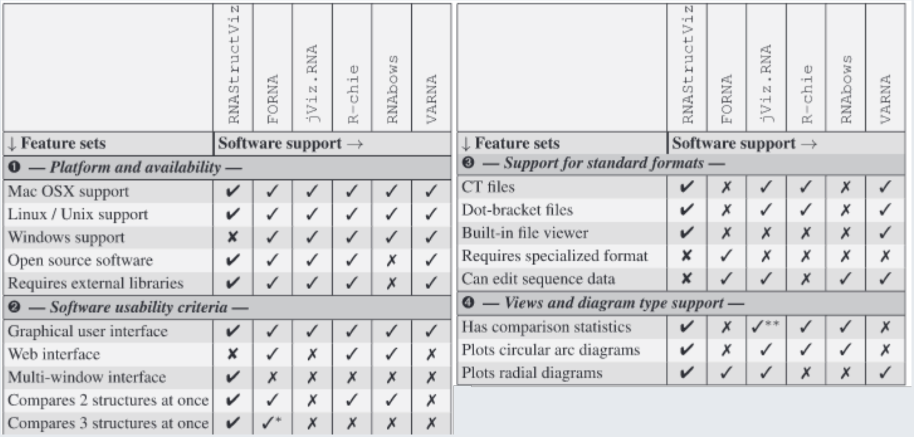
- Mathematical biology (abbreviated MathBio) is a field that uses mathematical models and theoretical abstractions of the structure of living organisms
- These models are used to explore biological principles dictating the underlying structures, the development and the behavior of these living systems
- In this talk, we will discuss my work as a software engineering RA with the *Georgia Tech Discrete Mathematics and Molecular Biology* (gtDMMB) research group led by Christine Heitsch
- N.b., my official title within the group is endearingly designated as the "*Code Goddess*" :)

## The RNAStructViz application

## RNAStructViz: Graphical base pairing analysis

- The historical RNAStructViz application was a project developed by Professor Christine Heitsch and her husband (Ph.D. in computer graphics from UCB) to visualize RNA secondary structures
- By the time I arrived and started work with gtDMMB in the summer of 2018, the old C++ source code was badly broken with modern Linux and MacOS compilers
- My work was to modernize the C++ source, add support for enhanced graphics using the `cairo` library, and to generally improve and support the project in the long term
- The key feature RNAStructViz provides is visualization and comparisons with arc diagrams

## RNAStructViz – Comparison of features

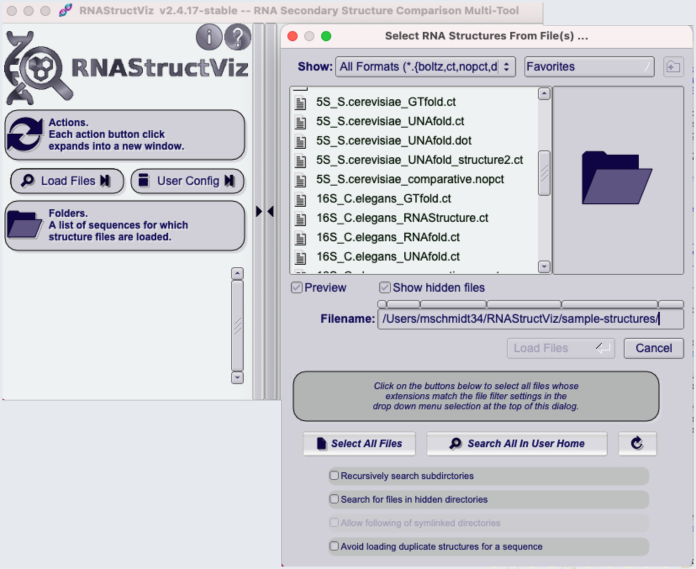


| ↓ Feature sets | RNAStructViz | FORNA | jViz.RNA | R-chie | RNAbows | VARNA |
|---|---|---|---|---|---|---|
| | Software support → | | | | | |
| ❶ — *Platform and availability* — | | | | | | |
| Mac OSX support | ✔ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Linux / Unix support | ✔ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Windows support | ✘ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Open source software | ✔ | ✓ | ✓ | ✓ | ✗ | ✓ |
| Requires external libraries | ✔ | ✓ | ✓ | ✓ | ✗ | ✓ |
| ❷ — *Software usability criteria* — | | | | | | |
| Graphical user interface | ✔ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Web interface | ✘ | ✓ | ✗ | ✓ | ✓ | ✗ |
| Multi-window interface | ✔ | ✗ | ✗ | ✗ | ✗ | ✗ |
| Compares 2 structures at once | ✔ | ✓ | ✗ | ✓ | ✓ | ✗ |
| Compares 3 structures at once | ✔ | ✓* | ✗ | ✗ | ✗ | ✗ |

| ↓ Feature sets | RNAStructViz | FORNA | jViz.RNA | R-chie | RNAbows | VARNA |
|---|---|---|---|---|---|---|
| | Software support → | | | | | |
| ❸ — *Support for standard formats* — | | | | | | |
| CT files | ✔ | ✗ | ✓ | ✓ | ✗ | ✓ |
| Dot-bracket files | ✔ | ✗ | ✓ | ✓ | ✗ | ✓ |
| Built-in file viewer | ✔ | ✗ | ✗ | ✗ | ✗ | ✓ |
| Requires specialized format | ✘ | ✓ | ✗ | ✗ | ✗ | ✗ |
| Can edit sequence data | ✘ | ✓ | ✓ | ✗ | ✓ | ✓ |
| ❹ — *Views and diagram type support* — | | | | | | |
| Has comparison statistics | ✔ | ✗ | ✓** | ✓ | ✓ | ✗ |
| Plots circular arc diagrams | ✔ | ✗ | ✓ | ✓ | ✓ | ✗ |
| Plots radial diagrams | ✔ | ✓ | ✓ | ✗ | ✗ | ✓ |

*A comparison of selected features across related tools; an extended survey appears in the RNAStructViz WIKI.*

## RNAStructViz Screenshot – Loading sample structures I


RNAStructViz v2.4.17-stable -- RNA Secondary Structure Comparison Multi-Tool

Select RNA Structures From File(s) ...

Show: All Formats (*.{boltz,ct,nopct,d | Favorites

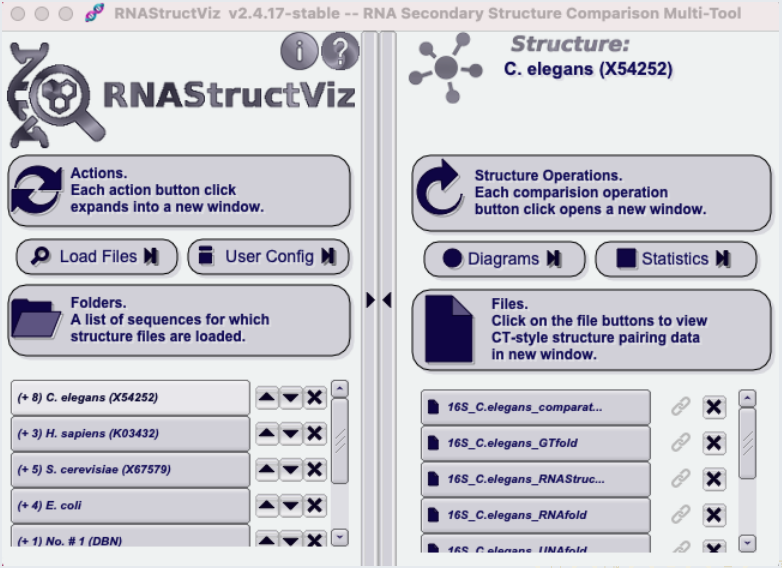
5S_S.cerevisiae_GTfold.ct
5S_S.cerevisiae_UNAfold.ct
5S_S.cerevisiae_UNAfold.dot
5S_S.cerevisiae_UNAfold_structure2.ct
5S_S.cerevisiae_comparative.nopct
16S_C.elegans_GTfold.ct
16S_C.elegans_RNAStructure.ct
16S_C.elegans_RNAfold.ct
16S_C.elegans_UNAfold.ct

Filename: /Users/mschmidt34/RNAStructViz/sample-structures/

*Click on the buttons below to select all files whose extensions match the file filter settings in the drop down menu selection at the top of this dialog.*

RNAStructViz
Actions.
Each action button click expands into a new window.
Load Files
User Config
Folders.
A list of sequences for which structure files are loaded.
Preview
Show hidden files
Load Files
Cancel
Select All Files
Search All In User Home
Recursively search subdirctories
Search for files in hidden directories
Allow following of symlinked directories
Avoid loading duplicate structures for a sequence

## RNAStructViz Screenshot – Loading sample structures II


RNAStructViz v2.4.17-stable -- RNA Secondary Structure Comparison Multi-Tool

Structure:
C. elegans (X54252)

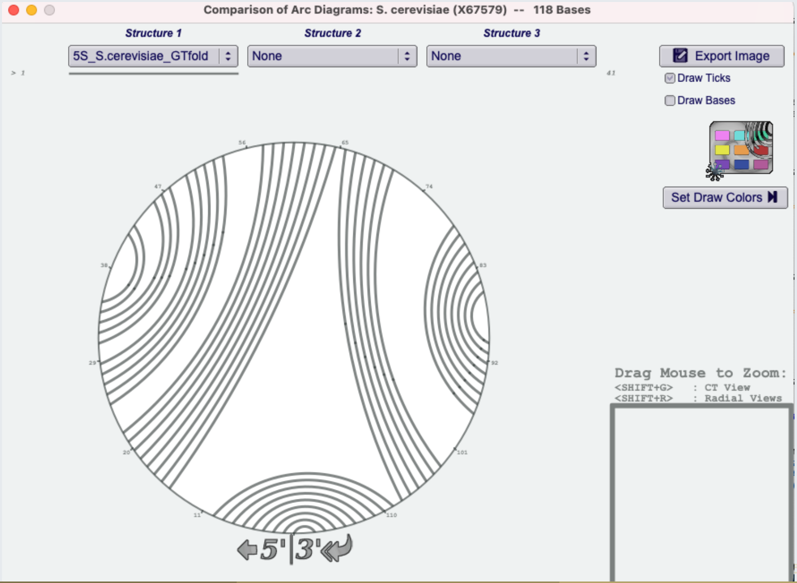
Actions.
Each action button click expands into a new window.

Structure Operations.
Each comparision operation button click opens a new window.

Folders.
A list of sequences for which structure files are loaded.

Files.
Click on the file buttons to view CT-style structure pairing data in new window.

## RNAStructViz Screenshot – Arc diagram window


Comparison of Arc Diagrams: S. cerevisiae (X67579) -- 118 Bases

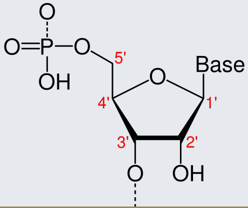
Structure 1: 5S_S.cerevisiae_GTfold
Structure 2: None
Structure 3: None

## Arc diagram window – Discussion

- The bases indexed from position #1 to `#LengthOfBaseSequenceString` are placed at equidistant spacings around a circle
- The sequentially numbered base pairs are ordered around the circle counter-clockwise starting from the position at 270 degrees, or $\frac{3\pi}{2}$ radians, labeled by the $5' \mid 3'$ directional arrows in the display window
- An arc connecting paired bases is drawn within the circle

---

**Example:** *A furanose (sugar-ring) molecule with carbon atoms labeled using standard notation. The* $5'$ *is upstream, whereas the* $3'$ *is downstream – diagram taken from* `WP/Directionality_(molecular_biology)`*:*



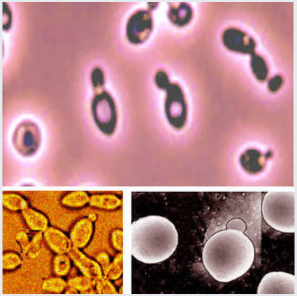
## Arc diagrams – Discussion example – S. Cerevisiae (yeast)

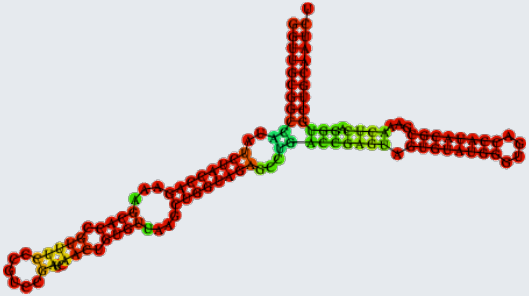
GGUUGCGGCCAUAUCUACCAGAAAGCACCGUUCCCGUCCGAUCAACUGUGUUAAGCUGGUAGA

((((((((....((((((((...((((.(((((......))..))).))))....))))))))

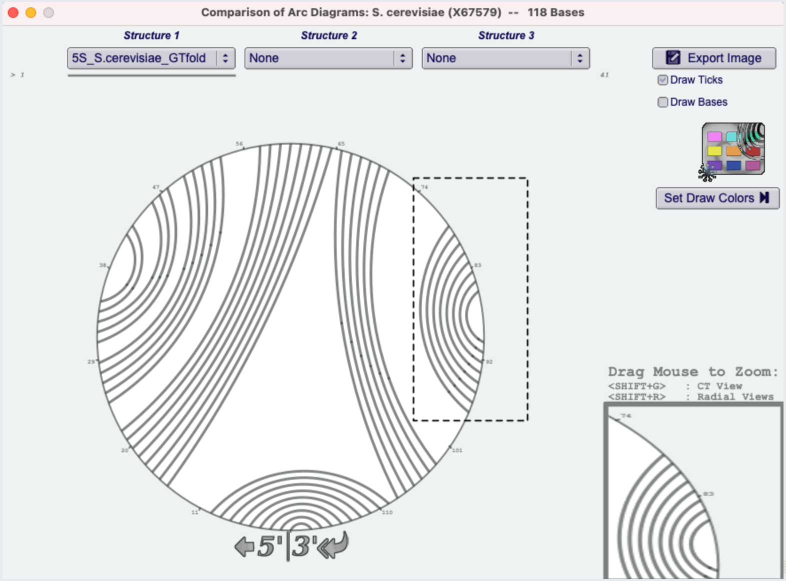
GCCUGACCGAGUAGUGUAUGGGUGACCAUACGCGAAACUCAGGUGCUGCAAUCU

.....(((((((.((((((((....))))))))...)))).))))))))))).


**(Actual microscopic views)**


**(Radial view of 2D MFE structure)**

## Arc diagram window – Zoom select

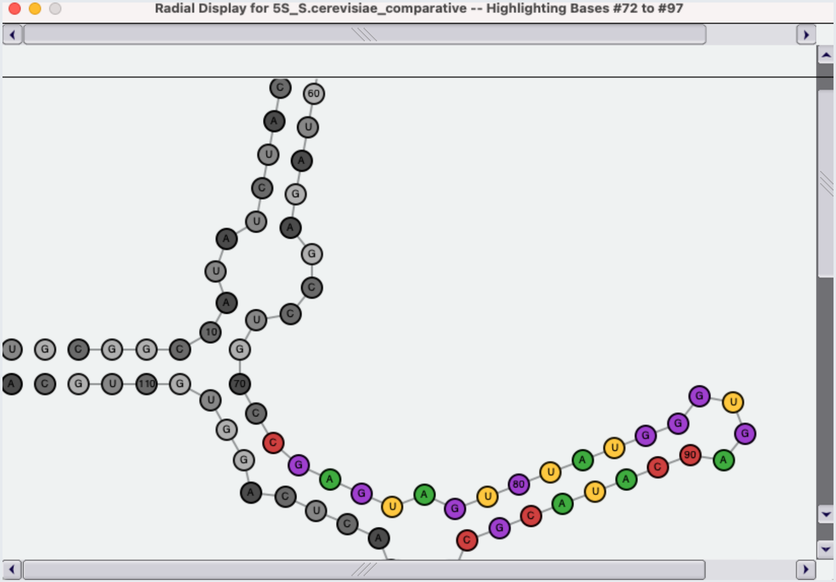

Comparison of Arc Diagrams: S. cerevisiae (X67579) -- 118 Bases

Structure 1: 5S_S.cerevisiae_GTfold
Structure 2: None
Structure 3: None

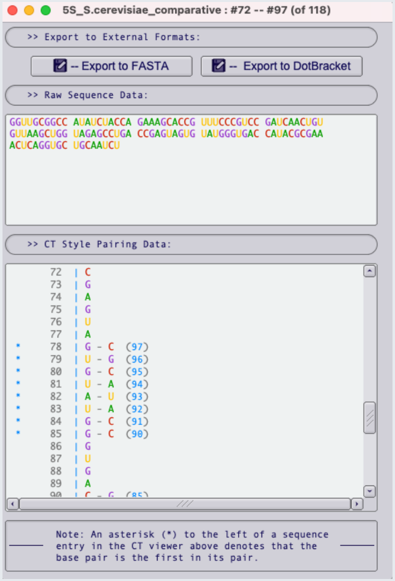
Export Image
Draw Ticks [✓]
Draw Bases
Set Draw Colors

Drag Mouse to Zoom:
<SHIFT+G> : CT View
<SHIFT+R> : Radial Views

## Arc diagram zoom – Radial layout visualization


Radial Display for 5S_S.cerevisiae_comparative -- Highlighting Bases #72 to #97

## Arc diagram zoom – CT segment visualization

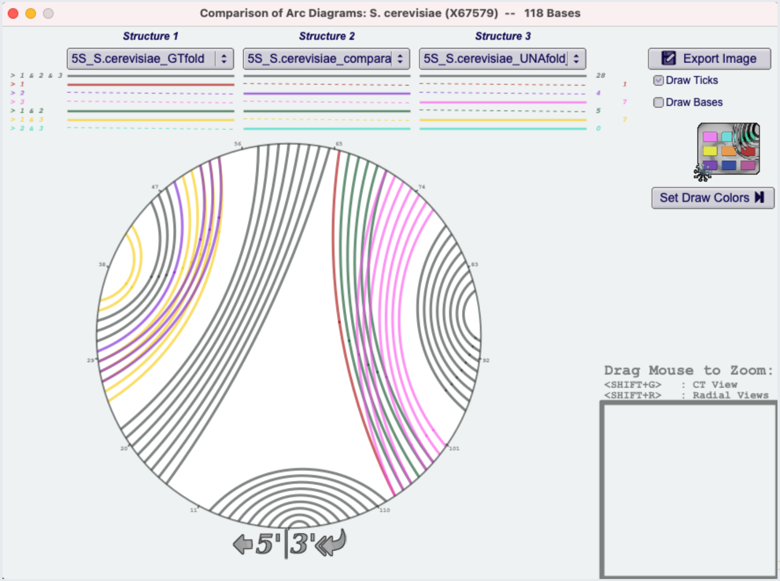

5S_S.cerevisiae_comparative : #72 -- #97 (of 118)
>> Export to External Formats:
-- Export to FASTA
-- Export to DotBracket
>> Raw Sequence Data:
>> CT Style Pairing Data:

```
GGUUGCGGCC AUAUCUACCA GAAAGCACCG UUUCCCGUCC GAUCAACUGU
GUUAAGCUGG UAGAGCCUGA CCGAGUAGUG UAUGGGUGAC CAUACGCGAA
ACUCAGGUGC UGCAAUCU
```

```
      72  | C
      73  | G
      74  | A
      75  | G
      76  | U
      77  | A
*     78  | G - C  (97)
*     79  | U - G  (96)
*     80  | G - C  (95)
*     81  | U - A  (94)
*     82  | A - U  (93)
*     83  | U - A  (92)
*     84  | G - C  (91)
*     85  | G - C  (90)
      86  | G
      87  | U
      88  | G
      89  | A
      90  | C - G  (85)
```

Note: An asterisk (*) to the left of a sequence entry in the CT viewer above denotes that the base pair is the first in its pair.

## Arc diagram window – Comparing multiple structures

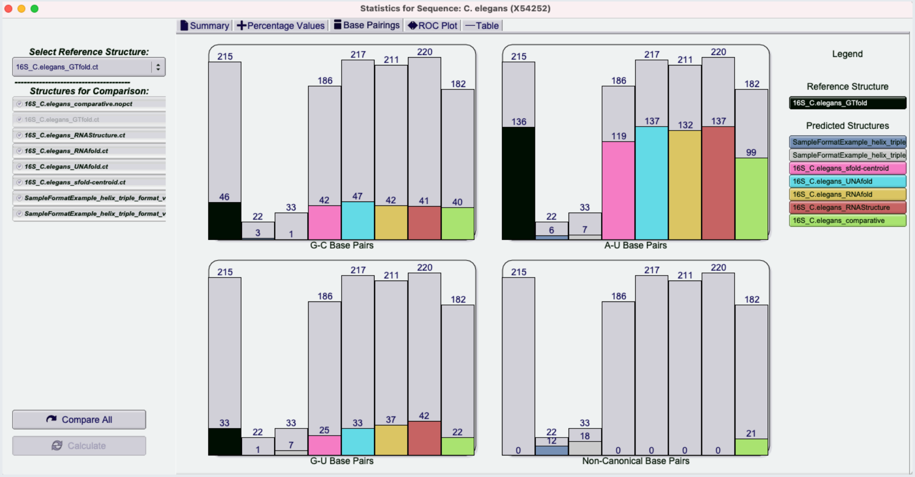


Comparison of Arc Diagrams: S. cerevisiae (X67579) -- 118 Bases

| | Structure 1 | Structure 2 | Structure 3 | | |
|---|---|---|---|---|---|
| | 5S_S.cerevisiae_GTfold | 5S_S.cerevisiae_compara | 5S_S.cerevisiae_UNAfold | | |
| > 1 & 2 & 3 | ━ | ━ | ━ | 28 | |
| > 1 | ━ | ╌ | ╌ | | 1 |
| > 2 | ╌ | ━ | ╌ | 4 | |
| > 3 | ╌ | ╌ | ━ | | 7 |
| > 1 & 2 | ━ | ━ | ╌ | 5 | |
| > 1 & 3 | ━ | ╌ | ━ | | 7 |
| > 2 & 3 | ╌ | ━ | ━ | 0 | |

Drag Mouse to Zoom:
<SHIFT+G> : CT View
<SHIFT+R> : Radial Views

## RNAStructViz Screenshot – Statistics window



Statistics for Sequence: C. elegans (X54252)

Summary | Percentage Values | Base Pairings | ROC Plot | Table

**Select Reference Structure:**

16S_C.elegans_GTfold.ct

**Structures for Comparison:**

- 16S_C.elegans_comparative.nopct
- 16S_C.elegans_GTfold.ct
- 16S_C.elegans_RNAStructure.ct
- 16S_C.elegans_RNAfold.ct
- 16S_C.elegans_UNAfold.ct
- 16S_C.elegans_sfold-centroid.ct
- SampleFormatExample_helix_triple_format_v
- SampleFormatExample_helix_triple_format_v

Compare All

Calculate

Legend

Reference Structure

- 16S_C.elegans_GTfold

Predicted Structures

- SampleFormatExample_helix_triple
- SampleFormatExample_helix_triple
- 16S_C.elegans_sfold-centroid
- 16S_C.elegans_UNAfold
- 16S_C.elegans_RNAfold
- 16S_C.elegans_RNAStructure
- 16S_C.elegans_comparative

## GTFoldPython software project

## GTFold – Overview and algorithms

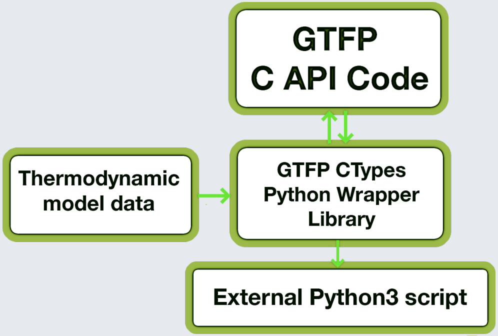
- Accurate and efficient RNA secondary structure prediction remains an important open problem in computational molecular biology
- GTFold is the first implementation of RNA secondary structure prediction by thermodynamic optimization for modern multi-core computers
- The difference in the parallel prediction program used by GTFold to take full advantage of today's modern computing technology is particularly valuable to researchers working with very RNA sequences, such as RNA viral genomes

## GTFold – Overview and algorithms II

- The decade old historical source code for GTFold produced several command line only utilities that could be used to generate secondary structures (especially, MFE and MFE structures)
- We were motivated by the need to run Python3 on a webserver to index sequences and corresponding computational data generated with GTFold
- One of my ideas as a gtDMMB graduate research assistant in the fall of 2019 was to write Python3 bindings around the original GTFold sources in C++
- Perhaps the most impressive slide of source code in the project is it's highly robust and cross platform war-tested `Makefile` and supporting set of install scripts :)

## GTFoldPython – Introduction

- The GTFoldPython (GTFP) project provides Python3 bindings based around the historical GTFold sources in C++
- The backend uses the Python3 C API
- The frontend is a wrapper library that uses `CTypes` to call the C API functions from a dynamic shared library (e.g., Windows DLL / MacOS Dylib / Linux SO)



## GTFoldPython – Comparison – Python C API code

```
PyObject * GetMFEStructure(const char *baseSeq, ConsListCType_t consList, int consLength) {
      /* Error checking omitted ... */
      MFEStructRuntimeArgs_t rtArgs;
      InitMFEStructRuntimeArgs(&rtArgs);
      rtArgs.baseSeq = baseSeq;
      SetRTArgsSequenceLength(rtArgs, strlen(baseSeq));
      if(ParseGetMFEStructureArgs(consList, consLength, &rtArgs) != GTFPYTHON_ERRNO_OK) {
           FreeMFEStructRuntimeArgs(&rtArgs);
           return ReturnPythonNone();
      }
      if(InitGTFoldMFEStructureData(&rtArgs) != GTFPYTHON_ERRNO_OK) {
           FreeMFEStructRuntimeArgs(&rtArgs);
           return ReturnPythonNone();
      }
      double mfe = ComputeMFEStructure(&rtArgs);
      if(GetLastErrorCode() != GTFPYTHON_ERRNO_OK) {
           return ReturnPythonNone();
      }
      if(WRITEAUXFILES) {
           ConfigureOutputFileSettings();
           save_ct_file(outputFile, baseSeq, mfe);
      }
      char *dbMFEStruct = ComputeDOTStructureResult(rtArgs.numBases);
      PyObject *mfeTupleRes = PrepareMFETupleResult(mfe, dbMFEStruct);
      Free(dbMFEStruct);
      FreeMFEStructRuntimeArgs(&rtArgs);
      FreeGTFoldMFEStructureData(rtArgs.numBases);
      if(mfeTupleRes == NULL) {
           return ReturnPythonNone();
      }
      return mfeTupleRes;
```

## GTFoldPython – Comparison – Wrapper library code

```
        ## Library initialization code:
        if GTFPConfig.PLATFORM_DARWIN:
            GTFoldPython._libGTFoldHandle = ctypes.cdll.LoadLibrary("GTFoldPython.dylib")
        else:
            GTFoldPython._libGTFoldHandle = ctypes.PyDLL("GTFoldPython.so",
                                                mode=ctypes.RTLD_GLOBAL, use_errno=True)
        @staticmethod
        def _WrapCTypesFunction(funcname, restype=None, argtypes=None):
            return GTFoldPython._libGTFoldHandle.__getattr__(funcname)

        @staticmethod
        def GetMFEStructure(baseSeq, consList = []):
            """Get the MFE and MFE structure (in DOTBracket structure notation)
            :param baseSeq: A string of valid bases (ATGU/X)
            :param consList: A list of constraints on the MFE structure
            :return: A tuple (MFE as double, MFE structure as string in DOTBracket notation)
            :rtype: tuple
            """
            GTFoldPython._ConstructLibGTFold()
            resType = ctypes.py_object
            argTypes = [ GTFPTypes.CStringType,
                         GTFPTypes.FPConstraintsListType(consList),
                         ctypes.c_int ]
            libGTFoldFunc = GTFoldPython._WrapCTypesFunction("GetMFEStructure", resType, argTypes)
            (mfe, mfeStruct) = libGTFoldFunc(GTFPTypes.CString(baseSeq),
                                             GTFPTypes.FPConstraintsList(consList),
                                             len(consList))
            return (float(mfe), str(mfeStruct))
```

## GTFoldPython – Example – Find MFE and MFE structure

**External Python3 script source:**

```python
import sys, os
from GTFoldPythonImportAll import *

GTFP.Init()
GTFP.Config(quiet = False, debugging = False, verbose = False, stdmsgout = "stderr")

baseSeqFPCons = "GCAUUGGAGAUGGCAUUCCUCCAUUAACAAACCGCUGCGCCCGUAGCAGCUGAUGAUGCCUACAGA"
consListFP = GTFPUtils.ReadFPConstraintsFromFile("../Testing/TestData/tRNA/yeast.fa.cons")

(mfe, mfeDOTStruct) = GTFP.GetMFEStructure(baseSeqFPCons, consListFP)
print("MFE␣%1.3f␣=>␣MFE␣DOT␣STRUCT␣\"%s\"\n\n" % (mfe, mfeDOTStruct))
```

---

**Terminal output printed upon invoking the script above:**

```
MFE -17.200 => MFE DOT STRUCT "(((((((((.........))))...........(((((.......)))))....)))))......
```

## Thermodynamic models – Introduction I

- Standardized sets of empirically well tested nearest neighbor (NN) parameters can be used to predict the stability of RNA secondary structures (and hence liklihood of a given structural variant to exist naturally)
- Common sets of these parameters, such as the NNDB models – Turner1999 and Turner2004 – e.g., those documented in [7, 1] – exist and are publicly posted for download on the pioneering researchers' websites
- There are also programs for generating modified sets of these parameters to use in computational simulation for comparison against benchmarked data:
  E.g., see `https://github.com/gtDMMB/GTModify`

## Thermodynamic models – Introduction II

- **Key idea:** Model RNA folding as a *dynamic system* of structures which represent the states of the system at a given time
- After a very long time, asymptotically, we expect that a given base sequence $\mathcal{S}$ will form every possible structure $\mathcal{P}$
- For each sequence $\mathcal{S}$ and possible structure $\mathcal{P}$, there is an associated probability for observing $\mathcal{S}$ folding to $\mathcal{P}$ at any given time
- The idea is to use models from mathematics and thermal physics to aim to define the most *reasonable* probabilities: $\mathbb{P}[\mathcal{P} \mid \mathcal{S}]$

## Thermodynamic models – The Boltzmann distribution I

- Let $\mathcal{X} := \{X_1, X_2, \ldots, X_N\}$ denote a system of states where the $i^{th}$ state, $X_i$, has an associated energy $E_i$
- We say that the system $\mathcal{X}$ is *Boltzmann distributed with temperature $T$* if and only if

$$\mathbb{P}[X_i] = \exp\left(-\beta E_i\right) \cdot Z^{-1}, \text{ for } Z := \sum_{i \geq 1} \exp\left(-\beta E_i\right), \beta := \left(k_B T\right)^{-1}$$

- The empirical value of the constant $k_B \approx 1.38 \times 10^{-23}$ is called the (classical) *Boltzmann constant*
- The parameter $\beta$ is called the *inverse temperature* of the system

## Thermodynamic models – The Boltzmann distribution II

- The Boltzmann distribution is usually attained in limiting *thermodynamic equilibrium* that occurs after letting the system progress for a long time period
- Why the dependence on a fixed temperature $T$ above?
  1. At very high temperatures we assume that all states are approximately equally probable to occur
  2. Whereas at very low temperatures we assume that only the “best“ states are likely to occur
- Why is the Boltzmann distribution important here? We can give a proof of the following information theoretic fact:
  *The Boltzmann distribution (BD) makes the least number of assumptions on the model (requires the least external / extraneous information). That is, formally, the BD is the unique distribution with the lowest information content, or equivalently, with the maximal (Shannon) entropy.*

## Thermodynamic models – NN model – Definitions

- We define the *probability of a base pair* $(i, j)$ *for* $\mathcal{S}$ to be

$$\mathbb{P}[(i,j) \mid \mathcal{S}] := \sum_{(i,j)\in P} \mathbb{P}[P \mid \mathcal{S}]$$

- We can then derive (see [3]) a recursive NN prediction for individual base pairs in the optimal structures as follows:
  - Initialize: $N_{i,j} := 0$ for all $i - j \leq m$;
  - Recursively compute:

$$N_{i,j} = \max\left(N_{i,j-1}, \max_{i \leq k < j-m} N_{i,k-1} + N_{k+1,j-1} + 1\right)$$

- Note that here, the notation $N_{i,j}$ denotes the distance from base pair $(i, j)$ (at the $i^{th}$ position) to the next base it should be paired with

## Thermodynamic models – NN and dynamic programming

- We define the *partition function $Z_{\mathcal{P}}$ for the set of RNA structures $\mathcal{P}$ of $\mathcal{S}$* by

$$Z_{\mathcal{P}} := \sum_{P \in \mathcal{P}} \exp(-\beta E(P))$$

- Computational strategy: Compute the partition functions $Z_{\mathcal{P}_{ij}}$ recursively
- This computation can be made efficient through the use of dynamic programming

## GTFoldPython – Specifying custom thermodynamic models

- This functionality used to be available in GTFold by setting: `export GTFOLD_DATADIR=$(readlink -f /path/to/thermo/data)`

**External Python3 script source:**

```
import sys, os
from GTFoldPythonImportAll import *

GTFP.Init()
GTFP.Config(quiet = False, debugging = False, verbose = False, stdmsgout = "stderr")

## Configure the energy model and thermodynamic parameters
## NOTE: Paths relative to the current running script location
#GTFP.SetGTFoldDataDirectory("../Testing/ExtraGTFoldThermoData/GTFoldTurner04/")
#GTFP.SetThermodynamicParameters(TURNER04, None)
GTFP.SetThermodynamicParametersFromDefaults(TURNER04)

baseSeqFPCons = "GCAUUGGAGAUGGCAUUCCUCCAUUAACAAACCGCUGCGCCCGUAGCAGCUGAUGAUGCCUACAGA"
consListFP = GTFPUtils.ReadFPConstraintsFromFile("../Testing/TestData/tRNA/yeast.fa.cons")

(mfe, mfeDOTStruct) = GTFP.GetMFEStructure(baseSeqFPCons, consListFP)
print("MFE␣%1.3f␣=>␣MFE␣DOT␣STRUCT␣\"%s\"\n\n" % (mfe, mfeDOTStruct))
```

---

**Terminal output printed upon invoking the script above:**

```
MFE -14.770 => MFE DOT STRUCT "((((((((.........))))..........(((((.......)))))....)))))......
```

## Concluding remarks

The End

Questions?

Comments?

Feedback?

Thank you for your time!

## References I

M. Andronescu, A. Condon, D. H. Turner and D. H. Matthews. *The determination of RNA folding nearest neighbor parameters*. Methods Mol. Biol. **1097**, pp. 45–70 (2014). https://doi.org/10.1007/978-1-62703-709-9_3

*GTFoldPython software documentation.* https://github.com/gtDMMB/GTFoldPython/wiki

MIT Math 18.417 Course Lecture Notes. *RNA ensembles*. Available online: https://math.mit.edu/classes/18.417/Slides/rna-ensembles.pdf

*RNAStructViz software documentation.* https://github.com/gtDMMB/RNAStructViz/wiki

Schmidt, M. D., Kirkpatrick, A., and Heitch, C. *RNAStructViz: graphical base pairing analysis*. Bioinformatics **197** (2021). https://doi.org/10.1101/2021.01.20.427505

## References II

Swenson, M. S., Anderson, J., Ash, A., Gaurav, P., Sukos, Z., Bader, D. A., Harvey, S. C., and Heitsch, C. E. *GTfold: Enabling parallel RNA secondary structure prediction on multi-core desktops*. BMC Research Notes. 5(1): 341, 2012. `https://github.com/gtDMMB/gtfold`

D. H. Turner and D. H. Matthews. *NNDB: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure*. Nucleic Acids Research, Volume 38, D280–D282 (2010). `https://doi.org/10.1093/nar/gkp892`